Power Fault Tolerance
Technical Report (WIP)

Protocol Labs

July 27, 2017

Abstract

Byzantine Fault Tolerance (BFT) accounts for faults as the number of faulty nodes and is thus cumbersome to apply to many modern decentralized systems. We introduce the Power Fault Tolerance (PFT) model, which reframes BFT in terms of participants' *influence* over the outcome of a protocol, instead of the number of nodes. In PFT, n is the total power, and f is the fraction of power controlled by faulty or adversarial participants.

This work:

(a) provides a formal definition and properties for PFT;
(b) generalizes Byzantine Consensus (BC) protocols of different classes (*permissioned*, *permissionless*, and *federated*) into a single class of Power Consensus (PC);
(c) explores new directions for PC protocols, particularly for *blockchains*, and protocols that can detect and make progress during catastrophic network partitions;

Work in Progress. This is a work in progress Technical Report from Protocol Labs. Active research is under way, and new versions of this paper will appear. For comments and suggestions, contact us at research@filecoin.io

1 Byzantine Fault Tolerance and Consensus

Before formally defining the Power model, we must review Byzantine Fault Tolerance (BFT) and Byzantine Consensus (BC).

Definition 1.1. (BFT) A (n, f)-BFT protocol has n participants and is able to tolerate up to f Byzantine faults. Traditionally, $n - f$ participants are *correct* (or honest, altruistic) and follow the protocol correctly; f are *faulty* (or byzantine, malicious, and adversarial), and may deviate from the protocol arbitrarily. Secure BFT protocols satisfy the following criteria:

- (Safety) If $n - f$ correct participants execute the protocol correctly, then any actions of f faulty participants cannot cause the protocol to fail unexpectedly.
- (Liveness) Correct participants *eventually* make progress.

There are many kinds of BFT protocols; this work is most relevant to *Consensus* and closely-related protocols:

- (BC) In *Byzantine Consensus* protocols, also known as *Byzantine Agreement* [4, 5, 9], participants propose and agree upon values in a sequence of epochs. This is one of the main academically studied classes of BFT protocols. Often, results are framed in terms of a binary version, *Binary Byzantine Agreement* (BBA).
- (BSR) In *Byzantine State-Machine Replication* protocols [1, 2, 13], participants receive a sequence of state-changing requests from clients, participants must propagate the effect of requests to each other, and all correct participants must come to agreement on the values or responses externalized to the clients.
- (BBC) In reliable *Byzantine Broadcast* protocols, a single designated *sender* (or dealer) sends out a value to the rest of the participants, and all correct participants must agree on the value. This is similar to but simpler than consensus. BBC is the original *Byzantine Generals* problem [9].

- (Blockchain) In *Blockchain* protocols [3, 6, 11, 12], a set of participants receive transactions from clients, and participants create a probabilistically consistent log of all transactions. Secure Blockchains can be constructed to be equivalent to BSR protocols.

The Power model is useful in all these variants. This work will primarily explore BC and Blockchain. The BC problem is characterized by the *propose* and *decide* events: every party executes $propose(v)$ to start the protocol and $decide(v)$ to terminate it with a value v.

Definition 1.2. (BC) A protocol for Byzantine Consensus (BC) with n players and up to f faults, who *propose* values, find agreement, and *decide* on values, satisfies:

- (Validity) If all correct parties *propose* value v, then some correct party eventually *decides* v.
- (Agreement) If some correct party *decides* on v, and some other correct party *decides* on v', then $v = v'$.
- (Termination) Every correct party eventually *decides*.

BC protocols are often structured in a sequence of *rounds* or *epochs*. At the end of a BC protocol, participants output a sequence of values $\mathcal{V}$, and at each epoch t, participants *decide* on a value $v_*^t \in \mathcal{V}$. During an epoch t, a single or multiple participants propose a single or multiple candidate values v_i^t, which are communicated to other participants the network. Participants vote on or commit to candidate values. Consensus for epoch t is achieved and v_*^t is agreed-upon when a candidate value v_i^t gathers enough (non-repudiable and non-faulty) commitments to pass a fault-tolerance threshold $n - f$, usually a fixed parameter of the protocol. Some protocols proceed in *non-overlapping* epochs, meaning a candidate value v_i^t is proposed and consensus is reached on v_*^t before a candidate value v_i^{t+1} for the next epoch $t + 1$ is proposed; other protocols proceed in *overlapping* epochs, where candidate values v_i^{t+1} may be proposed before consensus is achieved on v_*^t. Blockchains are most similar to sequential, epoch-overlapping BC.

1.1 Power in Consensus

The standard consensus models BC and BSR use networks of equal participants and model fault tolerance as a fraction of the total number of participants (e.g. $n \geq 3f + 1$). In a sense, every participant has an equal amount of power and influence over the outcome of the protocol. In this *permissioned* model, all participants must be known and authorized, otherwise an adversary could generate Sybil identities and trivially capture the consensus.

Modern open-membership or *permissionless* consensus protocols permit unknown participants to join the protocol, and thus must ensure Sybil identities confer no advantages to an adversary. This has been achieved in a variety of ways, yielding different classes of protocols:

- In *Proof-of-Work* Blockchain protocols [8, 11, 14], participants use control over some scarce resource (power) – such as computation, memory capacity, storage, or bandwidth – and commit this scarce resource as a vote on one of the potential outputs of the protocol. The fault-tolerance assumption is described in terms of a fraction of the total resources committed by the participants.
- In *Proof-of-Stake* Blockchain protocols [3, 7], participants accrue stake (power) by participating correctly in the protocol, and commit stake on one of the potential outputs of the protocol; stake is scarce, and can be reduced or entirely destroyed if faulty behavior is detected. The fault-tolerance assumption is described in terms of a fraction of the total stake of all participants.
- In FBA (*Federated Byzantine Agreement*) protocols [10], participants use quorum-slices – individual trust decisions, proportional to relative power, that determine system-level quorums. Fault-tolerance depends on quorum-slices and the individual trust decisions of participants (i.e. the influence participants exert on others).

The departure from simple participant counts toward several different types of counting majorities (e.g. resources, stake, relative trust, etc) has made it difficult to reason about all consensus protocols cohesively. It has also yielded many protocols that are tightly coupled to their specific type of consensus instead of being described generically. The *Power Fault Tolerance* model (PFT) unifies all these classes of protocols

by modeling the influence participants have over the output of the protocol as *power*, and recasting the traditional fault-tolerance assumptions in terms of total power n and a tolerated faulty fraction of power f.

2 Formalizing Power and Influence

Formalizing the notion of *power* helps us draw useful conclusions about consensus protocols across these different classes, as well as describe general protocols that work with any instantiation of power.

Definition 2.1. (Power) Protocol Π has total power $\mathcal{P}^t$ at each epoch t. A participant i has power p_i^t at epoch t, such that $\sum_i p_i^t = \mathcal{P}^t$

Remark. In some protocols, the total power is fixed across all epochs. In others, it changes across epochs as participants join or leave, or as participants acquire or lose power.

Definition 2.2. (Influence) The influence (or normalized power) $\mathcal{I}_i^t$ of a participant i over the output of epoch t is defined as the fraction of the total power $\mathcal{P}^t$ controlled by i, such that $\mathcal{I}_i^t = \frac{p_i^t}{\mathcal{P}^t}$ and $\sum_i \mathcal{I}_i^t = 1$.

Definition 2.3. (Power Consensus) An $(n,\ f)$-PowerConsensus protocol with k participants has n total power, and is capable of tolerating up to f faulty power, satisfies:

- (Validity) If all correct power *proposes* v, then some correct power eventually *decides* v.
- (Agreement) All correct power that *decides* on a value, *decides* on the same value v.
- (Termination) All correct power eventually *decides*.
- (Conservation) If power is committed to a candidate value v_a^t then it cannot be committed to any other value v_b^t, and all power committed to candidate values in epoch t must be up to n power ($\sum_i p^{v_i^t} \leq n$).

2.1 Variants of Power Protocols

How protocols instantiate power will have implications on the properties of the protocols. For example:

- *Fixed or Variable Power.* In some protocols, the total power is fixed across all epochs. In others, it changes across epochs as participants join or leave, or as participants acquire or lose power.
- *Public or Private Power.* In some protocols, nodes must publicly announce their power to the rest of the network. For example, in common *Proof-of-Stake* systems power is accounted publicly. In others, participants do not have to disclose their power. For example, in some *Proof-of-Work* systems, participants do not have to disclose their power; they can commit it privately and externalize only partial information.
- *Exact or Estimated Power.* In some protocols, power is accounted exactly (for example, common *Proof-of-Stake* and BC protocols). In others, power is estimated based on the likelihood of events (for example, common *Proof-of-Work* protocols). This applies separately to individual power (p_i) and total power ($\mathcal{P}$): it is possible to have exact total power with only estimates of individual power. The exactness of the accounting may drastically affect the security of the protocol.
- *Verifiable Power.* In order to conserve power, some protocols will need to provide a mechanism for participants to prove that they own the power they announced, or malicious users can pretend to have more power than they really have.
- *Rational.* Some byzantine protocols have looser restrictions, such as the n power must be controlled by *rational* actors, not directly *honest* ones.

3 Generalizing with Power Schemes

BFT Protocols can be described or formulated generally in terms of power and a *power scheme* PS where

$$\mathsf{PS} = (\mathsf{SetPower}, \mathsf{CommitPower}, \mathsf{CountPower})$$

- $\{0,1\} \leftarrow \mathsf{SetPower}(p_i^t)$ is a function that adjusts i's power at epoch t to p_i^t.
  - In some protocols, the power is explicitly fixed as a network parameter (e.g. $1/(\#$ of participants) in permissioned protocols).
  - In other protocols this must be accounted for publicly and verifiably (e.g. Proof-of-Stake protocols generally need to account all available stake).
  - In other protocols the power is a secret value that changes at any time at the will of each participant (e.g. in *Proof-of-Work* protocols users may acquire more resources and add them to the network at any time).

- $\{0,1\} \leftarrow \mathsf{CommitPower}(p_{i,j}^t,\ v_j^t)$ is a function that commits $p_{i,j}^t$ some of i's power at epoch t to the candidate value v_j^t.
  - It is critical that power is scarce, and cannot be committed to multiple candidate values. For example if *Proof-of-Work* protocols do not adequately ensure the power is committed only to a single candidate value, then participants could attempt to pursue multiple simultaneous histories and perhaps break the consensus.
  - In some protocols, each participant's power must only be committed in totality to a single candidate value (e.g. permissioned protocols, and *Proof-of-Stake* protocols), doing otherwise is considered faulty.
  - In other protocols, a participant's power can be committed partially to more than one candidate value (e.g. in *Proof-of-Work* protocols users may commit any fraction of their power to any candidate value, though this is generally not useful).

- $p^{v_j^t} \leftarrow \mathsf{CountPower}(v_j^t)$ is a function that counts the amount of power committed at epoch t to a candidate value v_j^t. Participants use CountPower to sort candidate values and pick the winner.
  - Usually, the power committed to a candidate value must pass the fault-tolerance threshold of the epoch before it is accepted as the winning value (e.g. the majority of the power: $p^{v_j^t} > \frac{1}{2}n^t$)
  - This could be counted in terms of *influence* instead, which may be easier for some protocols ($I^{v_j^t} > 0.5$).
  - In most protocols, the power committed to a candidate value is fixed for the duration of the protocol and counts the same for all participants.
  - In other protocols, the power committed to a candidate value is relative (e.g. in FBA protocols trust is relative, which also makes influence relative).

3.1 Example: traditional *permissioned* BC protocols

In the traditional *permissioned* BFT protocols, each participant's power and influence are equal (e.g. $p_i^t = 1$ and $I_i^t = 1/n^t$). In some protocols, all the participants are fixed for the duration of the protocol. In other protocols participants may change, adjusting influence accordingly.

- SetPower is always $\mathsf{SetPower}(1)$, called at initialization (or whenever a participant joins).
- CommitPower always commits a participant's full power, and is called when a participant votes for (*prepares*, or *commits*) a value.
- CountPower counts the votes (commitments) for a candidate value. Some protocols have leaders and a single proposed value per epoch. Other protocols are leaderless or allow proposing multiple values

simultaneously. The count must pass the fault-tolerance threshold (e.g. $p^{v_j^t} > \frac{1}{2}n^t$ or $p^{v_j^t} > \frac{2}{3}n^t$) for a value to be output.

3.2 Example: *Proof-of-Work* consensus protocols

In *Proof-of-Work* protocols, each participant's power at a particular epoch is determined by however many resources they commit to computing *Proofs-of-work* on top of candidate values. A participant's influence is their power divided by the total resources the network as a whole commits to all candidate values. In many *Proof-of-Work* protocols, a given p_i^t is secret and hard to calculate exactly, but the whole network's $\mathcal{P}^t$ can be estimated closely enough to adjust fault-tolerance thresholds.

For example, in Bitcoin, p_i^t is the hashing power participant i commits to mining for next block on top of a blockchain head of epoch t (the candidate value); this is a secret value and hard to estimate. $\mathcal{P}^t$ is the sum of all hashing power in the network at epoch t; this can be estimated based on the rate of chain growth, and adjusted for by changing the *Proof-of-Work* target per block. Errors in this estimate slow or speed up the protocol.

- SetPower is usually individual and secret; it is set implicitly whenever a participant changes their amount of available resources (in Bitcoin, when a miner increases or decreases their hashing power).
- CommitPower is usually individual and secret; it happens implicitly whenever a participant starts computing expensive *Proofs-of-work* to find a candidate value for epoch $t+1$, on top of their chosen candidate value of the previous epoch t. Candidate values usually externalize the resources committed towards crafting them. In Bitcoin, CommitPower happens when a miner commits hashing power to mine on top of a last blockchain head; and only mined blocks externalize any of the power committed.
  - In some protocols, only winning values externalize the work performed (and thus the power committed) to the rest of the network. Other protocols attempt to surface more of the power committed, and count it toward the fault-tolerance threshold (e.g. GHOST, Ethereum, Filecoin).
  - Future improvement directions for this class of protocols include (a) better accounting of all the power committed, (b) committing resources without wasting them, and (c) committing resources without using them, via probabilistic methods.
- CountPower counts the resources expended in calculating a given candidate value, and all parent values in that history. In Bitcoin and other blockchain protocols, this is known as the length, weight, or quality of a blockchain (or in some protocols, the whole blocktree).

4 Future Work

This technical report aims at presenting a definition of power and influence and models faults in a distributed system in terms of power. Future work for this technical report include:

- Formalization of *conservation of power*: power must be conserved whenever participants cast their vote for a candidate value.
- Provide examples for Proof-of-Stake protocols and FBA protocols.

Acknowledgements

This work is the cumulative effort of multiple individuals within the Protocol Labs team, and would not have been possible without the help, comments, and review of the collaborators and advisors of Protocol Labs. Juan Benet and Nicola Greco conceived of the power model. Juan Benet formalized the Power Fault Tolerance and Power Scheme in collaboration with the rest of the team, who provided useful contributions, comments, review and conversations.

References

[1] C. Cachin. State machine replication with byzantine faults. In *Replication*, pages 169–184. Springer, 2010.

[2] M. Castro, B. Liskov, et al. Practical byzantine fault tolerance.

[3] P. Daian, R. Pass, and E. Shi. Snow white: Robustly reconfigurable consensus and applications to provably secure proofs of stake. Technical report.

[4] D. Dolev and H. R. Strong. Authenticated algorithms for byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.

[5] C. Dwork, N. Lynch, and L. Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM (JACM)*, 35(2):288–323, 1988.

[6] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse. Bitcoin-ng: A scalable blockchain protocol.

[7] A. Kiayias, A. Russell, B. David, and R. Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. Technical report.

[8] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford. Enhancing bitcoin security and performance with strong consistency via collective signing.

[9] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

[10] D. Mazieres. The stellar consensus protocol: A federated model for internet-level consensus.

[11] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[12] S. Park, A. Kwon, J. Alwen, G. Fuchsbauer, P. Gazi, and K. Pietrzak. Spacemint: A cryptocurrency based on proofs of space.

[13] L. Ren, K. Nayak, I. Abraham, and S. Devadas. Practical synchronous byzantine consensus. *arXiv preprint arXiv:1704.02397*, 2017.

[14] G. Wood. Ethereum: A secure decentralised generalised transaction ledger.